*Manuscript

# IRESE: an Intelligent Rare-Event Detection System Using Unsupervised Learning on the IoT Edge

Zaffar Haider Janjua, Massimo Vecchio, Mattia Antonini, and Fabio Antonelli[a]

[a] *OpenIoT research unit, FBK CREATE-NET, Italy.*

**Abstract**

In the recent years, a rapid growth of IoT devices has been observed, which in turn results in a huge amount of data produced from multiple sources towards the most disparate cloud platforms or the Internet in general. In a typical cloud-centric approach, the data produced by these devices is simply transmitted over the Internet, for consumption and/or storage. However, with the exponential growth in data production rates, the available network resources are becoming the actual bottleneck of this huge data flowing. Therefore, several challenges are appearing in the coming years, which are mainly related to data transmission, processing, and storage along the so-called *cloud-to-thing continuum*. In fact, one of the most critical requirements of several IoT applications is low latency, which often hinders raw data consumption to happen at the opposite endpoint with respect to its production. In the context of IoT data stream analytics, for instance, the detection of anomalies or rare-events is one of the most demanding tasks, as it needs prompt detection to increase its significance. In this respect, Fog and Edge Computing seem to be the correct paradigms to alleviate these stringent demands in terms of latency and bandwidth as, by leveraging on re-configurable IoT gateways and smart devices able to support the distribution of the overall computational task, they envisage to liquefy data processing along the way from the sensing device to a cloud endpoint. In this paper, we will present IRESE, that is a rare-event detection system able to apply unsupervised machine learning techniques on the incoming data, directly on affordable gateways located in the IoT edge. Notwithstanding the proposed approach enjoys the benefits of a fully unsupervised learning approach, such as the ability to learn from unlabeled data, it has been tested against various audio rare-event categories, such as gunshot, glass break, scream, and siren, achieving precision and recall measures above 90% in detecting such events.

*Keywords:* IoT gateway, Edge Computing, Fog Computing, embedded intelligence, machine learning, outlier detection.

*Email address:* {janjua, mvecchio, m.antonini, fantonelli}@fbk.eu (Zaffar Haider Janjua, Massimo Vecchio, Mattia Antonini, and Fabio Antonelli)

## 1. Introduction

Improved cost-effectiveness and miniaturization of sensing devices have increased their utility in various domains of daily human life such as healthcare, transport, education, agriculture, and security. In a typical IoT environment, these sensing devices are connected to the Internet and responsible to continuously sense their surroundings and then transmit data to a cloud station for further processing. In the past few years, an exponential growth in IoT devices have been observed in the form of smart products. Based on the context, these devices of various *varieties* produce a huge *amount* of data at varying *rates*. According to an estimate by *Cisco Global Cloud Index*, the data produced by a variety of data sources will reach to around *500 zettabytes* by 2019, whereas the internet infrastructure will be capable to handle *10.4 zettabytes* by that time [1]. Similarly, according to *CISCO Internet Business Solutions Group* the number of devices connected to the internet will reach around *50 Billion* by 2020 [2]. These factors *(variety, amount of data, and variable data rate)* have raised serious concerns in an IoT environment, which mainly relates to data transportation, data storage, data processing, and security. The first concern, *transportation* of data, needs a high-speed Internet, which can quickly and efficiently transmit data to the destination. The second concern, *storing* huge amount of data, needs cloud services and other necessary networking infrastructure. The third concern, data *processing*, is important to be handled because raw data is not meaningful and it is required to transform raw data into meaningful information [3]. The fourth concern, *security* must be addressed for critical applications in which IoT data can be stolen or intruders can attack the system. Cloud-based paradigms are widely used in IoT systems, in which the data is pushed to the cloud and after computations, the outcome is delivered back to the local system. However, due to the proliferation of IoT, an increased amount of data is produced at the edge of the network. The limited network bandwidth is unable to meet the requirements of low-latency transportation of data coming at a high speed [4]. Therefore, one can conclude that Cloud Computing alone is not efficient enough to handle the IoT generated data in the coming years [5]. Since data production at the edge of a network is increasing, an adequate choice is to perform the necessary processing on an edge device; near the source. The edge devices are becoming more powerful and resource friendly with optimal utilization of resources such as memory and energy.

Formally, in Edge Computing nomenclature, an edge device (*e.g.*, a gateway) is used to perform computation over data. In fact, Edge Computing approaches aims at performing data processing as close as possible to its source. Moreover, it is an effort to involve decentralized agents to perform necessary processing, which can reduce the burden on centralized processing units [6]. Edge Computing is particularity useful in time-critical applications, in which quick data processing is required with prompt response in a particular situation. For example, Boeing 787 generates around 5 Gigabyte of data in each second and it needs a large bandwidth to transmit this data, which is not realistic [4]. In such scenarios computing at the edge is crucial, as users may need very fast responses from the system.

Edge Computing has gained much attention in the recent years due to improved resources and increased processing power of an edge device. Today, Edge Computing is widely used in various applications such as smart home, smart city, smart health, and smart transportation.

In these applications, data is processed by an edge device such as a gateway to extract meaningful information from it and take necessary actions. IoT generate a data stream which contains patterns indicating several events of interest. Data stream analytics is done to discover interesting patterns hidden in a *disorganized* and *unbounded* data stream. In this work, we will present a model which relies on the edge device to perform data stream analytics for discovering interesting patterns. In fact, our objective is to detect those patterns which reflect the occurrence of a *rare-event* or outlier in an IoT data stream. We have used the term *rare-event* instead of *outlier* or *anomaly* due to our use-case, which we will introduce later while discussing the contributions of this work. In the following subsections, we will define the problem in the context of IoT data stream containing rare-events, afterward, we highlight the contributions of our work.

### *1.1. Problem Formulation*

In the context of data stream analytics, the most demanding task is to discover patterns reflecting short duration abrupt changes in a data stream which may indicate an unusual situation or event [7]. In literature, different terms are used for such short duration abrupt changes including rare-event, anomaly, or outlier. Summarizing various definitions of these terms given in the literature [8, 9, 10], we can formally define a rare-event as follows.

**Definition 1.** *A rare-event, or outlier, is as an observation (or set of few observations) which occurs infrequently and deviates or inconsistent with respect to other observations so much that it becomes suspicious to indicate an irregularity or an anomaly in the given set of observations.*

It is important to detect rare-events occurring in a data stream, as it may be helpful in detecting a potentially hazardous situation. For example, a microphone is deployed in an outdoor environment, receiving typical city-related sounds (*e.g.*, cars, horns, birds, etc.). All of a sudden, a siren is heard; this sound is very different from the background audio, and for this reason, it is considered as a rare-event with respect to its background environment. Consider another example, a vibration sensor is deployed on a machine located in an industrial plant to continuously measure the vibrations generated by its motor(s); when the machine will start malfunctioning, an abnormal vibration pattern may be registered by the attached sensor, representing this a rare-event in the context of normal working conditions of that machine.

Due to the *network bandwidth vs. data production rate* bottleneck, Cloud Computing has limitations in rare-event detection. Particularly, in time-critical applications, cloud-based paradigms may not be able to generate timely alerts for users. On the other hand, in Edge Computing, IoT data can be locally processed by an intelligent gateway. Hence an edge device may be able to quickly detect rare-events occurring in a data stream, directly generating prompter warnings/alerts and reducing network traffic. In this work, our primary focus is to use an edge device to analyze digitized data streams produced by IoT devices, to detect rare-events there occurring. For this purpose, we propose an intelligent rare-event

detection system suitable for the IoT Edge, which we called *IRESE*. The system uses machine learning techniques to detect rare-events occurring in the incoming data streams. Fig. 1 illustrates the overall concept of IRESE: the IoT devices continuously sense the environment, while an edge device (intelligent gateway) processes the incoming data streams with the goal of detecting rare-event instances and then transmit it to a cloud-based storage.

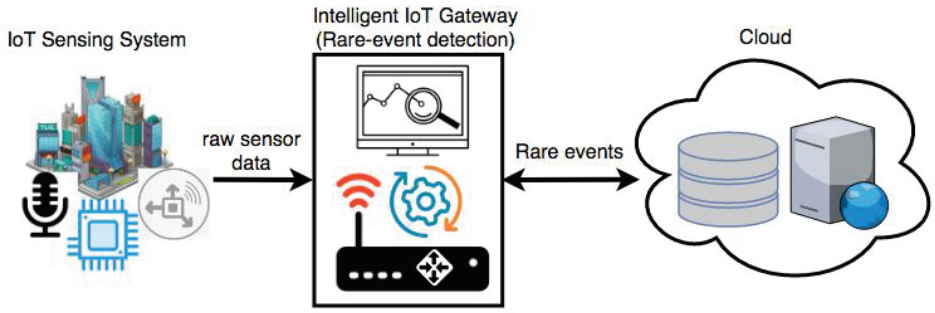


Figure 1: The conceptual diagram of IRESE.

### *1.2. Contributions*

Rare-event detection has been widely studied, resulting in a well-covered research area. However, IRESE strictly focuses on *edge device based rare-event detection* for IoT data streams. In Sec. 2, we will compare in detail the proposed approach with the most recent state-of-the-art research happening in this field. Briefly, the main contributions of this work can be summarized as following:

- Sharp technological focus on Edge Computing: in this paper, one of the major objectives is to utilize the typical computational power available on an edge device to process the data coming from various IoT devices. For this purpose, IRESE is deployed on an edge device. Such edge device is preferably an IoT gateway which continuously receives data from sensing devices and apply some data analytics techniques to detect rare-events. Edge computing helps to reduce the data transmission and analysis costs associated to a cloud-based IoT platform, as data is processed closer to its source and only the data patterns of interests are transferred to the upper layers (*i.e.*, from fog to cloud-level devices and infrastructures). Consequently, the cost (in terms of bandwidth and latency) of data transportation is consistently reduced. A recent and detailed review of edge-based IoT platforms is given in [11], while it is important to mention here that our goal is not to compare edge computing approaches against cloud computing ones, rather the aim is to introduce a framework which can empower edge devices to directly analyze data and discover useful patterns like rare-events. The motivation behind our choice to adopt an edge based approach is simply to reduce the burden of transmitting and processing data up to a cloud endpoint.

- Data stream analytics: we have strictly considered limitations of data stream analytics in the proposed model. Data streams are continuous, high speed, and unbounded.

Due to these unique characteristics, data stream needs a quick data processing without storing the data. For this reason, we have used data stream processing machine learning algorithms which work in two stages: micro-clustering and macro-clustering [12]. In a nutshell, micro-clustering enables an edge device to quickly get summaries of high speed incoming data stream in real-time without storing it, whereas macro-clustering further processes micro-clusters to discover separate clusters of rare-events and normal events. In order to demonstrated the effectiveness of IRESE, we have practically deployed it on a gateway device which continuously receives audio data through microphones and detect rare-events happened in an environment such as a gun shot. In fact, IRESE turns an edge device into an intelligent box, which continuously receives an audio stream and generates an alarm whenever a rare-event occurs.

- Detecting rare-event without prior knowledge: IRESE relies on a combination of unsupervised machine learning techniques. One of the challenging tasks in machine learning is to label data and provide it as training examples to a supervised machine learning algorithm. In order to avoid the efforts of having labeled data, we have considered unsupervised machine learning techniques, which do not need labeled data and once a rare-event is detected we can further investigated its type. Furthermore, unsupervised machine learning techniques allows us to automatically detect hidden pattern of interests in data, without having prior knowledge about these patterns. This feature is quite appealing to use it in an edge device as it is difficult to get knowledge about the raw data generated at source. In particular, we applied BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) algorithm [13] to get micro-clusters and Agglomerative Clustering [14] is used to get macro-clusters from the input data stream. More details on the applied techniques are presented in Section 3.4.

## 2. Related Works

Several Anomaly Detection (AD) techniques have been proposed in the literature using different machine learning approaches based on supervised, unsupervised, and semi-supervised training algorithms. Generally speaking, supervised learning techniques use training algorithms that require datasets with a sufficiently large number of instances. Then, in order to discriminate between normal events and anomalous ones, these datasets have to be labeled either manually or automatically. In this context, widely applied algorithms are multi-class Support Vector Machines (SVMs) [15], Bayesian classifiers [16], Neural Networks and Deep Neural Networks, Extreme Learning Machines [17], Gaussian Mixture Models (GMM) [18, 19], and Decision Trees [20].

Focusing on audio anomaly and rare event detection, several novel techniques have been proposed within the *Task 2* of the *DCASE 2017 Challenge* [21]. Many submitted techniques adopt deep neural network architectures [22, 23, 24, 25] to create classifiers able to detect the on-set time instant of rare-events (*e.g.*, gunshots, glass breaks, baby cries) over a background audio. However, supervised algorithms can be adopted if and only if a labeled dataset is available. Usually, these datasets are manually generated (*i.e.*, labeled) by researchers, but

this is an arduous and tedious work. Apart from being not affordable from time and money perspectives, this approach is not always feasible because some events are either extremely rare or unknown. For this reason, wherever possible, unsupervised approaches (*e.g.*, learning algorithms that can be trained using unlabeled datasets, because they are able to identify, extract, and learn patterns directly from data) are always advisable.

Oh et al. [26] propose an AD strategy based on an auto-encoder to detect audio anomalies produced by a Surface-Mounted Device (SMD) machine that places components on top of a Printed Circuit Board (PCB). The algorithm creates an auto-encoding manifold able to measure differences among instances and the manifold, signaling an anomaly if such distances are too large. Kouzumi et al. [27] propose a similar AD approach based on an auto-encoder. They trained the unsupervised algorithm by optimizing an objective function formulated by starting from the Neyman-Pearson lemma. In order to pursue this way, they assumed that the AD task was a statistical hypothesis test.

Recently, Bose et al. [17] proposed a novel approach to Anomaly Detection on the IoT Edge. There, the authors describe a new computing schema, called Anomaly Detection based Power Saving (ADEPOS), to adaptively update an anomaly detector, through time, without losing detection accuracy. The authors validated their approach by implementing a system to detect anomalies and failures of rotating bearing equipments by analyzing some time-based features extracted from vibrations. This technique consists in a group of one-class classifiers, which detect if an anomaly happened or not, followed by a majority voting strategy. ADEPOS is used to vary the number of detectors in the ensemble. Moreover, they evaluated the power saving of ADEPOS by simulating it in a Very Large Scale Integration (VLSI) hardware architecture. However, ADEPOS and IRESE have two different targets: the former aims to create adaptive anomaly detection systems, based on edge devices, that requires a small amount of energy. IRESE aims to create an Audio Rare-Event Detection system (Audio Anomaly Detection system), based on unsupervised machine learning, that runs on an IoT Gateway.

Another class of techniques that allow anomaly detection in audio streams are the semi-supervised learning algoritms. Aurino et al. [28] propose a 1-SVM approach within an automatic surveillance framework to detect burst-like audio events, namely screams, gun-shots, and glass breaks. Such an approach uses a two-stage classification scheme: the first stage classifies short audio segments (200 ms) through an ensemble of 1-SVM classifiers, while the second stage composes and re-classifies the first stage's decisions using a majority strategy, in order to take one decision per second. Elizalde et al. [29] present a framework to train audio event detectors using a semi-supervised self-training approach. Audio Event Detectors have to be firstly trained on the UrbanSound8K dataset [30], then have to run on unlabeled audio streams extracted from YouTube videos. If the detector recognizes a known sound with an high level of confidence, it is uses that sound to re-train the model. This approach helps to train models with acoustic diversity even if the original dataset is relatively small.

### *2.1. Audio Anomaly Detection in IoT contexts*

AD algorithms have been adopted also in IoT contexts, by creating more intelligent, reactive and secure environments. Hilal et al. [15] present and describe a Sensor Management framework called *IntelliSurv*. It realize an acoustic surveillance system that follows the pervasive IoT paradigm, being it able to detect and localize anomalous audio events using different kinds of distributed devices: smart sensors for environmental monitoring, and delegate sensors devoted to sensor management, localization and identification of anomalous events. Moreover, all the smart sensors have enough computing capabilities to locally execute the abnormality detection. At the classification stage of events, authors adopted SVM and LDA models.

Socoró et al. [19] propose an *Anomalous Noise Event Detector (ANED)* algorithm to map the traffic noise in urban and sub-urban areas using low-cost wireless sensor networks. These networks are composed of smart devices that perform simple signal pre-processing, then execute event detection using machine learning algorithms and finally they send labels to a central server that updates and draw noise maps. The authors there adopted a two-class classification scheme to distinguish the anomalous traffic noise (*e.g.*, jammed or semi-jammed traffic) from the normal traffic noise. They discovered that this approach performs better than the one using the one-class classifier, but they had to manually annotate the dataset. This system has been conceived using some outcomes from the European Project called DYNAMAP [31].

Alsina-Pagés et al. [32] present an Ambient Assisted Living (AAL) system, called homesound, that is able to detect and recognize different audio rare events happening in an everyday environment. This system uses a wireless sensor network to record audio from the environment; then the sensors forward the sampled audio streams to a GPU-based central device, which has two roles: first, it performs feature extraction from the raw audio stream, by computing 48 Mel Frequency Cepstral Coefficients (MFCC) and considering only the first 13 coefficients; then, it executes the inference of data using the trained model that is based on a classification algorithm (SVM) and clustering algorithm. The model response is finally sent to a remote system, where the medical staff can monitor the patient status.

## 3. Framework

In principle, the proposed model involves IoT devices which are deployed in an environment to measure signal energy through its transducer. An environment could be indoor or outdoor which has uniform characteristics and does not suppose to have frequent abrupt changes. We considered sensors which can produce a continuous waveform for the measured quantities such as acoustic events, vibrations, and acceleration. It is important that measured quantities are represented as a waveform as IRESE performs complex spectrum analysis to detect a rare-event. Fig. 2 shows the overall architecture of the rare-event detection system which is deployed on an edge device. IoT devices (for example, sensing devices) generate a data stream $D[n]$ sampled at sampling frequency $f_s$, where $f_s$ satisfies Nyquist-Shannon sampling theorem: $f_s \geq 2f_{max}$, $f_{max}$ represents the maximum frequency occurs

in the signal. An unbounded time series data stream is represented as a discrete signal: $D[n] = x_n, x_{n-1}, ..., x_{n-t}, ...$, where $x_n$ is the current sample, and $x_{n-t}$ is the first recorded sample. Since the data stream is unbounded, we need to buffer it to hold it for a small duration for further processing.

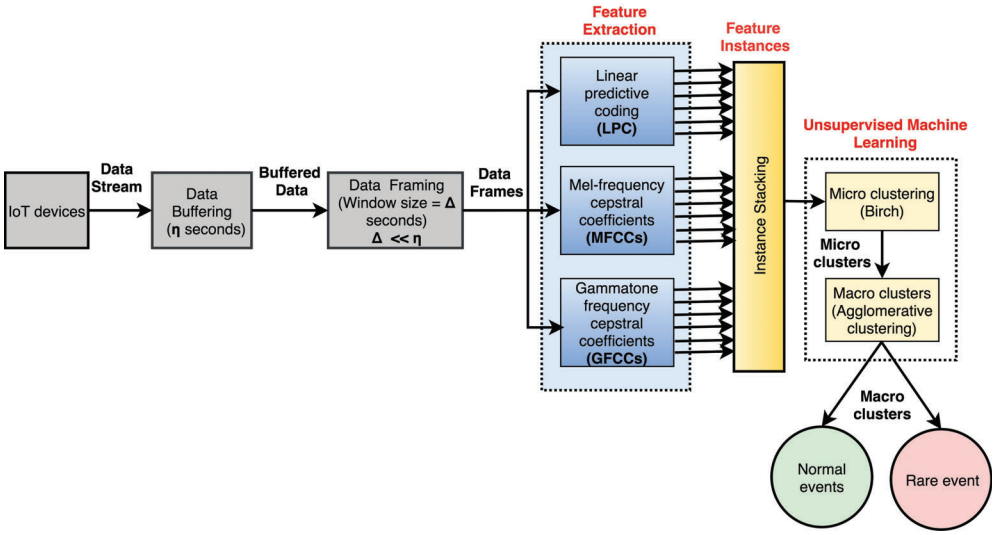


Figure 2: Proposed framework for rare-event detection system.

### *3.1. Data Buffering*

The incoming data stream $D[n]$ is periodically buffered in the local memory of an edge device. Each cycle is of fixed duration, in which data is buffered during a short interval of $\eta$ seconds, for example, 60 or 120 seconds. The data is buffered because it is continuously generated at a high speed, and buffering time allows IRESE to apply detection method on the buffered data. The buffering time $\eta$ could vary according to the type of data generated by IoT devices, however, it remains fixed for a particular setup. The buffered data is further supplied to a *Data Framing* module, which breaks it into even smaller frames which are suitable for feature extraction techniques.

### *3.2. Data Framing*

The data framing module takes buffered data and breaks it into smaller frames of duration $\Delta$ seconds, where $\Delta << \eta$, for example $\Delta$ is 1 second when $\eta$ is 60 seconds. For data framing, we defined a fixed length rectangular window of $\Delta$ seconds. The rectangular window function is represented in (1). It is a *tumbling* window, which moves over the buffered data stream in a way that two consecutive windows do not overlap with each other. For example, the buffer holds data for 60 seconds then data framing module breaks this buffered data into 60 equal sized frames by using a fixed window of size $\Delta = 1$ second.

$$\omega(n) = \begin{cases} 1 & \text{if } 0 \leq n \leq \Delta \cdot f_s \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

By multiplying the data stream $D[n]$ with the rectangular window function of (1), we obtain an individual frame $F_i[n]$, also represented as:

$$F_i[n] = D[n] \cdot \omega(n) \tag{2}$$

where, $F_i[n] = x_n, x_{n-1}, ..., x_{n-\Delta \cdot f_s}$ is the $i^{th}$ individual frame of buffered data, containing a sequence of samples selected during the interval starting at $n - \Delta \cdot f_s$ and ending at $n^{th}$ time instant.

### *3.3. Feature Extraction*

We have considered both time and frequency domain features to effectively and accurately detect abrupt changes visible in time or frequency domain. In order to preserve the time-domain envelope of the signal, we have used Linear Predictive Coding (LPC) [33], which is a well known technique used for feature extraction for audio and speech signals [34]. For the frequency domain analysis, we selected Mel-frequency cepstral coefficients (MFCCs) [35, 36] and Gammatone frequency cepstral coefficients (GFCC) [37]. MFCC and GFCC filter banks uniquely characterize the input signal to detect a rare-event. Thus, the feature vector $\nu$ is a tuple which is composed of subset features: $L_p(LPC)$, $M_f(MFCC)$, and $G_f(GFCC)$, which can be represented as $\nu = \{L_{p1}, L_{p2}, ..., L_{pi}, M_{f1}, M_{f2}, ..., M_{fj}, G_{f1}, G_{f2}, ..., G_{fk}\}$. In the following paragraphs, we will briefly explain these three types of feature extraction methods and also explain how we have used them in our model.

#### *3.3.1. Linear Predictive Coding (LPC)*

LPC [33] is a method which linearly combines past samples of a signal to predict its current sample. Exploiting the fact that speech and audio signals have redundancy, LPC is frequently used in such systems to detect various events. The algorithm is simple in which past samples are modulated as the weighted sum of $\rho$ previous values to minimize an error function. The error is actually the difference between actual samples and predicted samples. The weighted sum is estimated using coefficients of the error function. LPC algorithm recursively computes coefficients for each frame $F_i[n]$ in which the objective is to minimize the error $e_i[n]$ given in Equation 3. We have used auto-correlation [38, 34] method to compute LPC coefficients.

$$e_i[n] = F_i[n] - \hat{F}_i[n] \tag{3}$$

Where, $\hat{F}_i[n]$ is the predicted frame and $e_i[n]$ is the error.

#### *3.3.2. Mel-frequency cepstral coefficients (MFCCs)*

In order to include spectral analysis, we extracted MFCC [35, 36] for each frame $F_i[n]$. MFCCs has been widely used in various audio and speech recognition applications. The technique involves a series of steps: windowing (sub-framing), Discrete Fourier Transform *(DFT)*, computing Mel spectrum, computing log of Mel spectrum, and finally Discrete Cosine Transform (DCT) is computed to get Mel Frequency Cepstrum Coefficients (MFCCs).

An individual frame $F_i[n]$ will be the input of MFCC feature extraction block. The windowing function in MFCC technique breaks each frame into equal sized smaller sub-frames $s_r[n]$ of few milliseconds. In the end, *mean* value of all MFCCs is computed which are obtained from each sub-frame $s_r[n]$.

*3.3.3. Gammatone Frequency Cepstral Coefficients (GFCC)*

Although MFCC has been widely used for audio classification applications, it shows some limitations for signals having strong temporal domain signatures [37]. In [39, 40], authors have discussed such limitations of MFCC in time-frequency domain parameterization and feature selection methods. Considering such limitations, another biologically inspired technique has been proposed which is based on Gammatone (GT) filters [37]. The process of computing GFCC is more or less the same as of MFCC with the difference of using GT filer bank in GFCC. The GT filter bank is based on Gammatone function which is inspired from human auditory filter response [41]. The impulse response $g(t)$ of a GT filter is the product of Gamma distribution function and a sinusoidal tone having $f_c$ central frequency as given in Equation 4[37].

$$g(t) = Kt^{(n-1)}e^{-2\pi Bt}cos(2\pi f_c t + \varphi) \qquad t > 0 \tag{4}$$

where $K$ is the filter amplitude; $n$ is the filter order; $f_c$ is the central frequency in Hertz; $\varphi$ is the phase shift; and $B$ is the duration of impulse response. We have computed GTCC for each sub-frame $s_r[n]$. Like MFCC, an individual frame $F_i[n]$ is the input to GFCC block, which further divided into sub-frames $s_r[n]$. The GTCC is computed for each sub-frame $s_r[n]$ and, afterward, a *mean* is computed for all the GTCCs obtained from each sub-frames.

*3.4. Unsupervised Machine learning*

In Section 2, we have discussed in detail the application of machine learning in anomaly detection. However, in this work, our emphasis is to automatically extract patterns of rare-events occurring in an IoT data stream using an edge device. In case of supervised machine learning, we need to individually label patterns of rare-events exhibited by extracted features. Data labeling is a difficult and time consuming task as it requires an expert who closely observes incoming instances and assign them meaningful labels [42] In order to avoid the effort involved in data labeling and to automatically find the patterns of rare-events hidden in a data stream, we have used a two-stage rare-event detection strategy which relies on a combination of state-of-the-art unsupervised machine learning techniques. As shown in Figure 2, the unsupervised machine learning module takes stacked feature instances as input and process it in two stages to detect the occurrence of a rare-event. Here it is important to highlight the working of an unsupervised machine learning technique, which basically aims to partition data instances in a way that similar instances are grouped in same cluster [43]. Hence, dissimilar instances belong to different clusters. Exploiting the fact that the patterns of rare-events are reasonably different from the normal events, IRESE tries to find two separate clusters in the incoming data stream. Eventually, in these two clusters, one cluster contains instances of normal events where as the other cluster contains instances of rare-events.

The two-stage strategy is used due to the one-pass constraint of a high speed incoming data stream [44]. It is not possible to store such high speed data stream due to lack of resources and amount of data produced. The incoming data stream is processed in two-stages: online micro-clustering, and offline macro-clustering [12, 45]. In the first stage, online micro-clustering, the high speed data stream is processed in real-time to quickly extract statistical information from it in the from of micro-clusters. Micro-clusters could indicate the presence of rare-event patterns in the data steam. Therefore, it is further processed in the second stage *i.e.*, offline macro-clustering, which in an offline phase and extracts rare-events from the incoming data stream. As mentioned earlier, the final output is in the form of two clusters: *cluster A* is dense and containing data points reflecting normal behavior, whereas *cluster B* containing a rare-event (if it exists) which is an outlier and different from other events occurring in that specific interval of buffered data. A further detail of both stages is described in the following subsections.

*3.4.1. Micro-Clustering*

Since data streams are unbounded and having large amount of data, an efficient method is required to extract important statistics from the data in real-time. An online micro-clustering [12, 45] technique considers *one pass* nature of streaming data and attempts to quickly and efficiently collect the useful summary of data. One pass means that it is not sutiable to store raw data and it must be efficiently processed in first attempt to get meaningful information from it. The outcome of micro-clustering is several small clusters having unique properties due to the similarity between instances observed during the small time duration. There are several stream clustering techniques available for online micro-clustering which are compared in [45]. We have used the BIRCH (acronym of Balanced Iterative Reducing and Clustering using Hierarchies) algorithm, that is a tree-based stream clustering algorithm proposed in [13]. The algorithm constructs a *clustering feature (CF)* tree for incoming data instances, in which leaf nodes are micro-clusters. BIRCH is a fast and memory efficient algorithm and these characteristics make it suitable to be used in an edge device.

*3.4.2. Macro-Clustering*

In the offline macro-clustering phase [12, 45], micro-clusters are further processed and merged together to produce bigger clusters. The merging of clusters is based on the distance between the cluster centroids. Hence, the clusters having centroids close to each other are merged together to form a single cluster. Keeping in mind the fact that a rare-event has distinctive features, which keeps it in a separate cluster. We have used *Agglomerative Clustering* [44, 46] and used *Ward method* [14] to recursively merge micro-clusters by minimizing variance between them.

Fig. 3 shows the overall process of cluster merging. Following Ward algorithm, note that $d$ is the squared Euclidean distance between the centroids of any two given micro-clusters. A low value of $d$ shows that two micro-clusters are close to each other having similar characteristics, whereas a high values of $d$ means that two clusters are far from each other due to their varying characteristics. The algorithm recursively merges any two

given micro-clusters at each step while optimizing the *objective function* which is based on minimizing the total with-in cluster variance. The algorithm continuous the merging process until only two clusters left indicating the normal events and rare-events, if exist. The objective function considers a threshold value $Th$, which decides whether two micro-clusters are close enough to be merged together or not. Theoretically, increasing the value of $Th$ expands the size of a recursively merged cluster while reducing the detection rate of a rare-event, Whereas decreasing the value of $Th$ results in recognizing a normal event as a rare-event. The $Th$ value varies from one environment to another and it is selected after empirical analysis of received data.

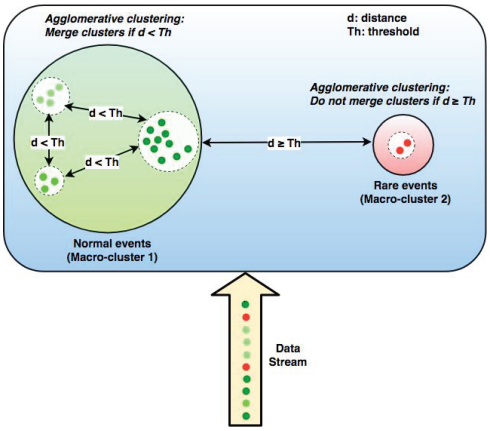


Figure 3: Macro-cluster formation in IRESE.

## 4. Experimentation

In order to quantitatively assess the proposed approach, we have conducted experiments with a typical use case involving the processing of audio data containing rare-events. In fact, we can safely extend our hypothesis that IRESE is also valid for other similar use cases which involve data streams from IoT devices having similar temporal and spectral characteristics. For example, another suitable scenario is the detection of faults in the machines using vibration and acoustic sensors. This section explains experiments conducted to detect various types of rare-events. Continuing the discussion from previous sections, the experiments validate the following claims in the context of rare-event detection: *i)* Detection of rare-events with high precision is the core objective of this work, as lower precision values generate more false alarms; *ii)* IRESE should independently detect rare-events, without considering the type of an event; *iii)* rare-events should be detected without having any prior knowledge since, as already mentioned, it is often difficult to develop prior knowledge on rare-events in the form of labels or experts' advice; *iv)* the whole process should be automated and scalable while considering the same features for other similar use cases; *v)* IRESE cannot be arbitrarily complex, as it is supposed to be executed by tiny edge devices.

*4.1. Experimental setup*

Connected devices and IoT technologies are spreading in all the application domains (*e.g.*, health-care, smart homes, wearable devices, etc.) and they are changing how humans interact with the surrounding environment. Typically, these devices have constrained computing and networking capabilities in order to reduce costs and energy consumption since are often battery-powered. In many scenarios, devices need an external entity, called gateway, that is deployed close to devices and it is able to execute computing- and networking-intensive operations, *e.g.*, bridging different networking worlds like Bluetooth and Wi-Fi. One of the players in the open-source landscape is the Adaptive Gateway for dIverse muLtiple Environment (AGILE)[47]. AGILE is a modular software framework for IoT gateways with a wide support for many network stacks and devices. Moreover, AGILE has been designed by following the micro-service paradigm that was initially conceived for distributed systems. This paradigm defines that all modules of the system are independently designed and implemented and they are able to interact among them using a well-defined set of Application Programming Interfaces (APIs). This enables strong modularity, resiliency against failures, scalability, reliability and simpler maintenance. The paradigm has been successfully applied to different domains (*e.g.*, Cloud Computing).

AGILE follows this paradigm in order to implement modules and services: all modules expose a set of APIs, via DBus[1] or RESTful interfaces, that enable interactions and data exchange. In this way, the gateway is more resilient against failures since, in the worst case, if a module crashes the system remains alive maybe with a limited set of capabilities. Finally, AGILE runs on x86 and ARM-based platforms like Raspberry Pi[2].

The framework for audio rare-event detection presented in this work has been implemented as an independent micro-service within the AGILE gateway framework. Since this module requires a raw audio stream in order to extract features, the AGILE gateway board, *i.e.*, a Raspberry Pi, is powered with a USB microphone. This microphone is recognized as a classic microphone by the gateway operating system. The micro-service records the audio stream from the microphone, then it performs data buffering and windowing. Thus, it extracts features over a temporal frame by computing MFCC, LPC and GFCC coefficients. Consequently, the module feeds the algorithm with the feature vector and finally verifies if the anomaly happened or not by checking which cluster, normality or anomaly, contains the audio frame.

This module can be used with two different data source: recorded (from microphone) audio stream and evaluation audio stream. Fig. 4 shows how it is possible to choose the data source. If we select the *recorded audio stream*, the system behaves as presented above. If we choose the latter stream, the module loads the audio stream from WAV files stored in the SD card of the gateway. Using this mode, we can evaluate the system performances as will be described in Section 4.

[1] `https://dbus.freedesktop.org/`

[2] `https://www.raspberrypi.org/`

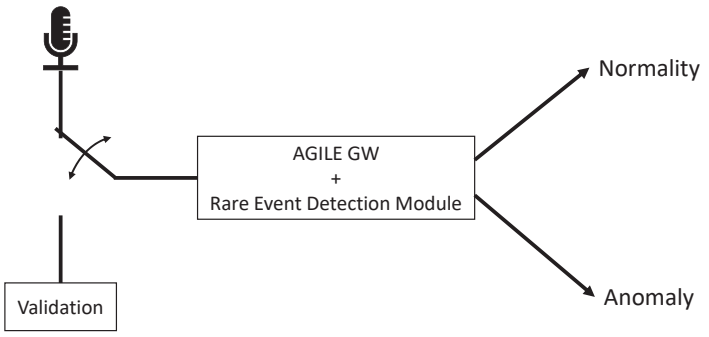


Figure 4: Functional schema, implementation of IRESE with AGILE.

*4.2. Software tools*

The overall system is implemented in Python. Three types of features are extracted using three python libraries: 1) LPC features are extracted using audiolazy[3] python library; 2) MFCC are extracted using librosa[4]; 3) GFCC are extracted using gammatone python library[5]. We have used scikit-learn[6] to apply unsupervised machine learning techniques Birch and Agglomerative Clustering.

*4.3. Dataset*

There are several datasets available for various audio events — UrbanSound8K [30], TUT Sound Events [48], and Audio set by Google [49] to name a few. In [50], author has provided a resourceful compilation of various audio datasets which include tagged and mixed audio events. Since we are using unsupervised machine learning to detect rare-events, we needed a dataset having labeled time stamps of various rare-events with normal background audio signals. In DCASE 2017 Challenge, authors produced a dataset with various backgrounds for three events: gunshot, glass break, and baby cry. However, in our understanding, their mixture model is not suitable for our case study, as we are looking for relatively more prominent rare-events from different sources and having varying characteristics which could highlight the seriousness of the situation For this purpose, we produced a dataset by mixing various rare-events, from multiple sources, with different backgrounds. In order to ensure the relatedness of this work with the state-of-the-art research happening in the domain, we rely on already published datasets to produce our mixture models. Therefore, we collected background sounds from DCASE 2017 Challenge dataset, and collected a subset, containing several variations, of rare-events from UrbanSound8K or downloaded directly from Freesound search engine. In particular, we have considered *four* types of rare-events: *gunshot, glass break, scream, and siren*. Furthermore, the sounds in each type of rare-event are also different from each other. We created in total 160 samples of each type of adding

[3] `https://pythonhosted.org/audiolazy/`
[4] `https://librosa.github.io/librosa/`
[5] `https://github.com/detly/gammatone`
[6] `http://scikit-learn.org/stable/`

a rare-event at a random time instant in a background sound. We used Pydub[7], python library, for mixing rare-events with background sounds. Each sound clip randomly contains exactly one rare-event. The sound is sampled at 44.1 KHz, which meets the standard audio sampling rate. Since we are simulating an IoT environment, we can safely assume that these sounds are similar as received by a microphone deployed in the environment. As illustrated in Fig. 2, the data received from the IoT devices is temporarily stored in a buffer for few seconds. The buffer size is variable, however, it remains fixed for a particular environment. In these experiments, we have considered the buffer size equals to 30 seconds, which is simulated by taking 30 second sound clip each time. The 30 second sound is further split into frames, and for each frame features are extracted. We already discussed in detail the feature extraction method in Sec. 3. However, here it is important to mention the number of coefficients, we have considered for each of three types of features extraction methods: LPC, MFCC, and GFCC. We have taken 10 coefficients of LPC, 40 MFCCs, and 40 GFCCs. Thus, in total, the length of the feature vector is 90 in which each value is a floating point.

*4.4. Experimental results*

In this section, we will explain empirical results obtained while conducting experiments using IRESE on the dataset described above. The two-staged unsupervised machine learning strategy of IRESE ultimately produces two clusters: a cluster of normal events, and a separate cluster of rare-event, if it exists. As mentioned in the previous section, we have synthetically constructed the dataset, in which we have added a rare-event sound at a random time instant in a background sound of relatively longer duration. For the sake of evaluation, we recorded the time instant at which we added a rare-sound in a background sound clip. The recorded information is used to evaluate the performance of IRESE by comparing the time instant, called "On-Set", at which IRESE detects a rare-event to the real time instant when the rare-event actually occurred according to records.

In order to evaluate the model, we have used matching matrix values: True positive (TP), False positive (FP), and False Negative (FN). In these experiments, a $TP$ occurs when IRESE correctly separates a rare-event observation from the rest of the observations. A FP occurs when IRESE wrongly detects a background sound or a normal event as a rare-event, whereas a FN occurs when IRESE fails to distinguish between a rare-event and background sounds. Additionally, we have also calculated precision (P), recall (R), and f-measure (F1) values, where $P$ gives us the positive predictive value, $R$ gives us true positive rate, and $F1$ score gives us the harmonic mean of $P$ and $R$ values. In conclusion, the value of $P$ decreases with an increase in number of $FP$ and, similarly, the value of value of $R$ decreases as number of $FN$ increases. Following equations are used to calculate these measures:

$$P = \frac{TP}{TP + FP} \tag{5}$$

$$R = \frac{TP}{TP + FN} \tag{6}$$

[7] `https://pydub.com`

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{7}$$

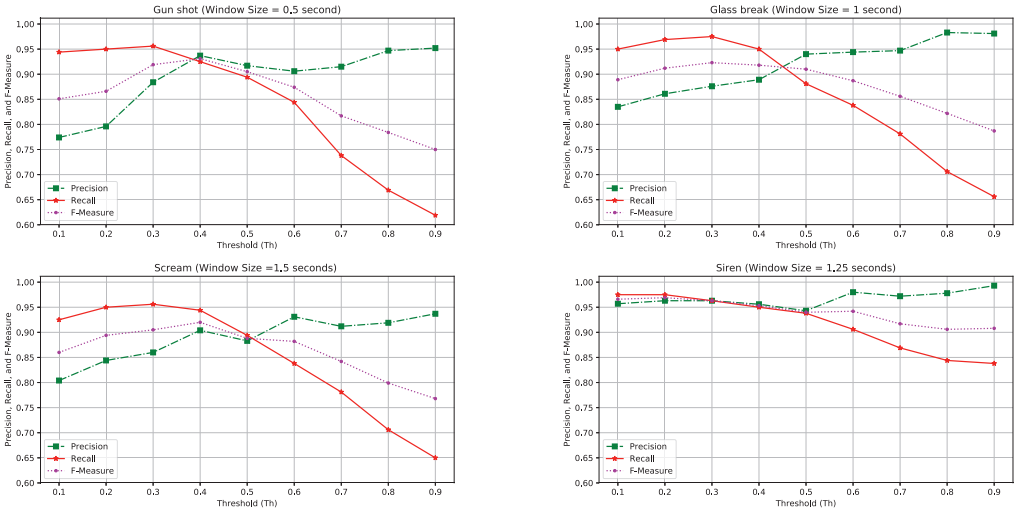


Figure 5: Finding the optimized threshold (Th) values to detect rare-events

Fig. 5 shows a plot of $P$, $R$, and $F1$ values against the threshold ($Th$) values discussed in Sec. 3.4.2 using a specific window size. It is clearly observable that as $Th$ value increases the precision increases and recall decreases. It confirms the trend that rare-event detection rate decreases with the increase in $Th$ value, whereas more false predictions are produced with low values of $Th$. The reason is that the boundary of the cluster defining normal events grows with the value of $Th$. Consequently, at a certain point, the size of the normal cluster grows so much that even an anomalous observation (occurring at a relatively larger distance) becomes part of normal cluster which increases the number of FN. We have selected an optimum value of $Th$, which could be observed in the graphs, where the combination of all three values ($P$, $R$, and $F1$) is highest. Thus, for gunshot the optimum value of $Th$ is 0.4 by using a window size of 0.5 second, for glass break the optimum value of $Th$ is 0.45 by using a window size of 1 second, for siren the optimum value of $Th$ is 0.3 by using a window size of 1.25, and for scream event the optimum value of $Th$ is 0.4

| | **Gun shot (Th=0.4)** | | | **Glass break (Th=0.45)** | | | **Siren (Th=0.3)** | | | **Scream (Th=0.4)** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Window Size | TP | FP | FN | TP | FP | FN | TP | FP | FN | TP | FP | FN |
| 0.25 | 151 | 21 | 9 | 156 | 36 | 4 | 156 | 11 | 4 | 157 | 46 | 3 |
| 0.5 | **148** | **10** | **12** | 153 | 32 | 7 | 155 | 11 | 5 | 157 | 28 | 3 |
| 1 | 143 | 11 | 17 | **149** | **12** | **11** | 154 | 6 | 6 | 157 | 20 | 3 |
| 1.25 | 125 | 27 | 35 | 136 | 18 | 25 | **154** | **6** | **6** | 151 | 19 | 9 |
| 1.5 | 116 | 28 | 44 | 124 | 27 | 36 | 152 | 7 | 8 | **151** | **16** | **9** |
| 2 | 103 | 26 | 57 | 102 | 23 | 58 | 152 | 6 | 8 | 145 | 13 | 15 |

Table 1: TP, FP, and FN results using IRESE for various rare-events with different window sizes.

Table 1 shows the values of TP, FN, and FP measures for the four types of events. Each row in the table represents the results obtained for a particular window size. Notice

that window size is the size of an individual frame, as defined in (1). We can observe a trend in values that TP decreases as window size increases, and it is true for all the cases. Consequently, FN increases as the window size increases, whereas FP does not follow a specific trend; it is probably due to using different background sounds which may contain some sounds similar to the rare-events.

| | **Gun shot (Th = 0.4)** | | | **Glass break (Th =0.45)** | | | **Siren (Th=0.3)** | | | **Scream (Th=0.4)** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Window Size | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| 0.25 | 0.87 | 0.94 | 0.91 | 0.81 | 0.97 | 0.88 | 0.93 | 0.97 | 0.95 | 0.77 | 0.98 | 0.86 |
| 0.5 | **0.93** | **0.92** | **0.93** | 0.82 | 0.95 | 0.88 | 0.93 | 0.96 | 0.95 | 0.84 | 0.98 | 0.91 |
| 1 | 0.92 | 0.89 | 0.91 | **0.92** | **0.93** | **0.92** | 0.96 | 0.96 | 0.96 | 0.88 | 0.98 | 0.93 |
| 1.25 | 0.82 | 0.78 | 0.8 | 0.88 | 0.85 | 0.86 | **0.96** | **0.96** | **0.96** | 0.88 | 0.94 | 0.91 |
| 1.5 | 0.8 | 0.72 | 0.76 | 0.82 | 0.77 | 0.79 | 0.95 | 0.95 | 0.95 | **0.9** | **0.94** | **0.92** |
| 2 | 0.79 | 0.64 | 0.71 | 0.81 | 0.63 | 0.71 | 0.96 | 0.95 | 0.95 | 0.91 | 0.9 | 0.91 |

Table 2: Performance evaluation results using IRESE for rare-event detection.

While looking at Table 2, we can estimate a suitable window size to detect a particular type of rare event and using an optimum value of $Th$. The precision increases as the number of FP decreases, whereas recall increases as number of FN decreases. In general, we can observe that suitable window size vary from one event to another and it depends on the duration of occurrence of a particular event. In summary, window size 0.5 seconds show the optimum detection performance with $precision = 0.93$, $recall = 0.92$, and $F\text{-}measure = 0.93$. For the glass break, the optimum window size is 1 second with a $precision = 0.92$, $recall = 0.93$, and $F\text{-}measure = 0.92$. Detection of sirens performs better with a window size of 1.25 seconds with all $precision$, $recall$, and $F\text{-}measure$ equals to 0.96. The highest suitable window size is observed for scream which is 1.5 seconds with a $precision = 0.9$, $recall = 0.94$, and $F\text{-}Measure = 0.92$.

| | **Gun shot** | | | **Glass break** | | | **Siren** | | | **Scream** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Window Size | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| 0.25 | 0.66 | 0.96 | 0.79 | 0.63 | 0.98 | 0.77 | 0.9 | 0.97 | 0.93 | 0.66 | 0.96 | 0.78 |
| 0.5 | **0.76** | **0.94** | **0.84** | 0.72 | 0.97 | 0.83 | 0.95 | 0.97 | 0.96 | 0.78 | 0.96 | 0.86 |
| 1 | 0.76 | 0.91 | 0.83 | **0.83** | **0.95** | **0.89** | 0.94 | 0.97 | 0.95 | 0.81 | 0.98 | 0.89 |
| 1.25 | 0.63 | 0.84 | 0.72 | 0.76 | 0.88 | 0.82 | **0.95** | **0.97** | **0.96** | **0.83** | **0.96** | **0.89** |
| 1.5 | 0.64 | 0.81 | 0.72 | 0.73 | 0.86 | 0.79 | 0.96 | 0.95 | 0.95 | 0.79 | 0.92 | 0.85 |
| 2 | 0.68 | 0.74 | 0.71 | 0.71 | 0.78 | 0.75 | 0.96 | 0.95 | 0.95 | 0.8 | 0.9 | 0.84 |

Table 3: Performance evaluation results without using IRESE (only macro-clustering) for rare-event detection.

In our understanding, this variation in optimum window sizes is due to the duration of rare-events. For example, a gunshot sound is sudden and exists for a very short duration such as between 0.5 seconds to 1 second. On the other hand, the sound of scream normally last longer (upto few seconds) such as 1.5 seconds or 2 seconds, which is also obvious from the results.

In order to prove the significance of IRESE, we have also calculated the results of rare-event detection using only Agglomerative Clustering technique (*i.e.*, macro-clustering stage). Note that two-stage strategy, micro-clustering followed by macro-clustering, improves the rare-event detection rate, which is obvious by comparing the results presented in Table 2 and Table 3. While using IRESE, we can see an improvement in all three calculated ($P$, $R$, and $F1$ ) values for different rare-events with different window sizes. Besides this improvement, the major benefit we achieve with IRESE is its suitability for deploying it in an edge device.

The micro-clustering stage of IRESE is able to quickly extract the statistical information from an incoming high speed data stream, without storing the data. Later on, this statistical information is further processed to make macro-clusters, which eventually indicates the presence of rare-events in the incoming data stream. Hence, IRESE is an effort to provide a solution to detect rare-events without storing incoming data on an edge device and it also empowers an edge device with artificial intelligence to reduce the burden on a cloud; sending only the patterns of interest to the cloud.

## 5. Conclusion

Edge Computing is becoming crucial with the increase in the amount of data produced from various IoT devices. Due to the limited amount of network resources, data processing near the source is highly required, especially for time critical applications. Moreover, the data stream generated by multiple sources always contain interesting patterns, which must be discovered to take important decisions. In this context, rare-event detection using an edge device is a promising research area, in which the objective is to detect critical events quickly and near the source, so that necessary actions can be taken accordingly. The proposed system, IRESE, has shown a significant performance to detect various types of rare-events: gunshot, glass break, siren, and scream. Moreover, we have used two-staged unsupervised machine learning strategy, which enable the system to detect interesting patterns in the form of rare-events without having any prior knowledge. The two-tier architecture considers one-pass nature of data streams and quickly extracts statistical information using micro-clustering and, afterward, micro-clusters are recursively merged to separate rare-events from the normal events. We have practically implemented and tested the whole system using an AGILE-based IoT gateway, which allowed us to fine tune IRESE's parameters based on the actual hardware limitations. In conclusion, IRESE is a lightweight and portable system, which could be quickly and easily deployed in various environments and start detecting rare-events with initiating any training session.

## Acknowledgements

This research was funded by the AGILE project, within the Horizon 2020 programme of the European Union, grant number 688088.

## References

[1] C. V. Networking, Cisco global cloud index: Forecast and methodology, 2015-2020. white paper, Cisco Public, San Jose.

[2] D. Evans, The internet of things: How the next evolution of the internet is changing everything, Cisco Internet Business Solutions Group (IBSG) 1 (2011) 1–11.

[3] R. L. Ackoff, From data to wisdom, Journal of Applied Systems Analysis 16 (1989) 3–9.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646.

[5] B. Zhang, N. Mor, J. Kolb, D. S. Chan, N. Goyal, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, J. Kubiatowicz, The cloud is not enough: Saving iot from the cloud, in: Proc. of the 7th USENIX Conference on Hot Topics in Cloud Computing, 2015, pp. 21–21.
[6] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (5) (2016) 78–81.
[7] A. I. Rana, G. Estrada, M. Sol, V. Munts, Anomaly detection guidelines for data streams in big data, in: 2016 3rd International Conference on Soft Computing Machine Intelligence (ISCMI), 2016, pp. 94–98.
[8] D. M. Hawkins, Identification of outliers, Vol. 11, Springer, 1980.
[9] V. Barnett, T. Lewis, Outliers in statistical data, Wiley, 1974.
[10] D. Moore, G. McCabe, B. Craig, Introduction to the Practice of Statistics, W.H. Freeman, 2009.
[11] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (1) (2018) 450–465.
[12] C. C. Aggarwal, S. Y. Philip, J. Han, J. Wang, A framework for clustering evolving data streams, in: Proceedings 2003 VLDB Conference, Elsevier, 2003, pp. 81–92.
[13] T. Zhang, R. Ramakrishnan, M. Livny, Birch: An efficient data clustering method for very large databases, SIGMOD Rec. 25 (2) (1996) 103–114.
[14] F. Murtagh, P. Legendre, Wards hierarchical agglomerative clustering method: which algorithms implement wards criterion?, Journal of classification 31 (3) (2014) 274–295.
[15] A. R. Hilal, A. Sayedelahl, A. Tabibiazar, M. S. Kamel, O. A. Basir, A distributed sensor management for large-scale IoT indoor acoustic surveillance, Future Generation Computer Systems 86 (2018) 1170–1184.
[16] L. Koc, T. A. Mazzuchi, S. Sarkani, A network intrusion detection system based on a hidden nave bayes multiclass classifier, Expert Systems with Applications 39 (18) (2012) 13492–13500.
[17] S. K. Bose, B. Kar, M. Roy, P. K. Gopalakrishnan, A. Basu, Adepos: Anomaly detection based power saving for predictive maintenance using edge computing, in: Proceedings of the 24th Asia and South Pacific Design Automation Conference on - ASPDAC19, ACM Press, 2019.
[18] J. Montalvão, D. Istrate, J. Boudy, J. Mouba, Sound event detection in remote health care - small learning datasets and over constrained gaussian mixture models, in: 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 1146–1149.
[19] J. Socoró, F. Alías, R. Alsina-Pagès, An anomalous noise events detector for dynamic road traffic noise mapping in real-life urban and suburban environments, Sensors 17 (10) (2017) 2323.
[20] J. Kevric, S. Jukic, A. Subasi, An effective combining classifier approach using tree algorithms for network intrusion detection, Neural Computing and Applications 28 (S1) (2016) 1051–1058.
[21] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, T. Virtanen, DCASE 2017 challenge setup: Tasks, datasets and baseline system, in: Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017), 2017, pp. 85–92.
[22] H. Lim, J. Park, Y. Han, Rare sound event detection using 1D convolutional recurrent neural networks, Tech. rep., DCASE2017 Challenge (September 2017).
[23] E. Cakir, T. Virtanen, Convolutional recurrent neural networks for rare sound event detection, Tech. rep., DCASE2017 Challenge (2017).
[24] W. Kaiwu, Y. Liping, Y. Bin, Audio events detection and classification using extended R-FCN approach, Tech. rep., DCASE2017 Challenge (September 2017).
[25] F. Vesperini, D. Droghini, D. Ferretti, E. Principi, L. Gabrielli, S. Squartini, F. Piazza, A hierarchic multi-scaled approach for rare sound event detection, Tech. rep., DCASE2017 Challenge (September 2017).
[26] D. Oh, I. Yun, Residual error based anomaly detection using auto-encoder in SMD machine sound, Sensors 18 (5) (2018) 1308.
[27] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, N. Harada, Unsupervised detection of anomalous sound based on deep learning and the neymanpearson lemma, IEEE/ACM Transactions on Audio, Speech, and Language Processing 27 (1) (2019) 212–224.
[28] F. Aurino, M. Folla, F. Gargiulo, V. Moscato, A. Picariello, C. Sansone, One-class svm based approach

for detecting anomalous audio events, in: 2014 International Conference on Intelligent Networking and Collaborative Systems, 2014, pp. 145–151.
[29] B. Elizalde, A. Shah, S. Dalmia, M. H. Lee, R. Badlani, A. Kumar, B. Raj, I. Lane, An approach for self-training audio event detectors using web data, in: 2017 25th European Signal Processing Conference (EUSIPCO), 2017, pp. 1863–1867.
[30] The urban sound dataset, `https://urbansounddataset.weebly.com/urbansound8k.html`, last accessed: April 16, 2019.
[31] The dynamap project web site, `http://www.life-dynamap.eu/`, last accessed: April 16, 2019.
[32] R. Alsina-Pagès, J. Navarro, F. Alías, M. Hervás, homeSound: Real-time audio event detection based on high performance computing for behaviour and surveillance remote monitoring, Sensors 17 (4) (2017) 854.
[33] D. O'Shaughnessy, Linear predictive coding, IEEE potentials 7 (1) (1988) 29–32.
[34] J. D. Markel, A. J. Gray, Linear prediction of speech, Vol. 12, Springer Science & Business Media, 2013.
[35] P. Bansal, S. A. Imam, R. Bharti, Speaker recognition using mfcc, shifted mfcc with vector quantization and fuzzy (2015).
[36] S. Davis, P. Mermelstein, Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Transactions on Acoustics, Speech, and Signal Processing 28 (4) (1980) 357–366.
[37] X. Valero, F. Alias, Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification, IEEE Transactions on Multimedia 14 (6) (2012) 1684–1689.
[38] J. Makhoul, Linear prediction: A tutorial review, Proceedings of the IEEE 63 (4) (1975) 561–580.
[39] K. Umapathy, S. Krishnan, S. Jimaa, Multigroup classification of audio signals using time-frequency parameters, IEEE Transactions on Multimedia 7 (2) (2005) 308–315.
[40] S. Chu, S. Narayanan, C.-C. J. Kuo, Environmental sound recognition with time–frequency audio features, IEEE Transactions on Audio, Speech, and Language Processing 17 (6) (2009) 1142–1158.
[41] B. R. Glasberg, B. C. Moore, Derivation of auditory filter shapes from notched-noise data, Hearing research 47 (1-2) (1990) 103–138.
[42] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.
[43] Z. Ghahramani, Unsupervised learning, in: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), ML Summer Schools 2003: Advanced Lectures on Machine Learning, Springer Berlin Heidelberg, 2004, pp. 72–112.
[44] S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O'Callaghan, Clustering data streams: Theory and practice, IEEE transactions on knowledge and data engineering 15 (3) (2003) 515–528.
[45] M. Carnein, D. Assenmacher, H. Trautmann, An empirical comparison of stream clustering algorithms, in: Proceedings of the Computing Frontiers Conference, ACM, 2017, pp. 361–366.
[46] L. Rokach, O. Maimon, Clustering methods, in: Data mining and knowledge discovery handbook, Springer, 2005, pp. 321–352.
[47] The agile project website, `https://agile-iot.eu/`, last accessed: April 16, 2019.
[48] The tut audio dataset, `http://arg.cs.tut.fi/index.php/datasets`, last accessed: April 16, 2019.
[49] The audio dataset by google, `https://research.google.com/audioset/`, last accessed: April 16, 2019.
[50] T. Heittola, Audio datasets, `http://www.cs.tut.fi/~heittolt/datasets`, last accessed: April 16, 2019.